Towards a Framework for One-sided RDMA Multicast

Xin Zhe Khooi
National University of Singapore
Singapore

Cha Hwan Song
National University of Singapore
Singapore

Mun Choon Chan
National University of Singapore
Singapore

ABSTRACT

We present the design and prototyping of a framework to support multicast for remote direct memory accesses (RDMA), specifically the one-sided WRITE operation. We use P4 programmable hardware to augment fixed-function RDMA transport hardware found on commodity NICs to enable one-sided RDMA multicast with zero-CPU overhead. Finally, we outline the potential challenges and future directions in realizing the framework for large-scale data center deployments.

CCS CONCEPTS

• **Networks → Data center networks**; **Programming interfaces**.

KEYWORDS

RDMA, multicast, programmable switches, SmartNICs, P4

ACM Reference Format:
Xin Zhe Khooi, Cha Hwan Song, and Mun Choon Chan. 2021. Towards a Framework for One-sided RDMA Multicast. In *Symposium on Architectures for Networking and Communications Systems (ANCS '21), December 13–16, 2021, Layfette, IN, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3493425.3502766

1 INTRODUCTION

RDMA is a technology that allows end hosts in the network to directly exchange data in the main memory and offload network I/O responsibilities from the CPU to the RDMA-capable network interface cards (RNICs). RDMA offers the potential of exceptional performance for high-performance systems [9, 18, 28] and thus the data center network landscape has been gradually shifting towards RDMA [9, 10].

There are two communication paradigms in RDMA, e.g., one-sided operations and two-sided operations. For one-sided operations, two communicating end hosts first exchange their connection parameters, i.e., queue pair number (QPN), packet sequence number (PSN), remote access key (RKEY), and remote virtual address (RADDR). With the learned connection parameters, one host can then perform READ, WRITE, and ATOMIC operations on the remote hosts main memory directly. This process incurs zero-CPU overhead on the remote end. On the other hand, two-sided operations – SEND and RECV, do not operate directly on the remote end's main memory, and thus no RKEY and RADDR is required. Instead, the CPU is actively involved [14] in buffering, and packets are processed upon arriving at the userspace similar to other kernel bypass techniques [8, 11].

With the rapid growth of network traffic in data centers and the end of Moore's law looming, the efficient usage of network and compute resources has been the utmost priority for hyperscalers [10]. To optimize network resources, multicast presents a promising option for the plethora of data center applications [19] that exhibit one-to-many group communication patterns, e.g., file system replication [26], distributed coordination [17], and virtual tenant intra-networking [25]. Multicast also accelerates application performance [5, 27, 29]. As to conserve the precious CPU resources, network I/O operations can be offloaded to dedicated hardware like RNICs. Unfortunately, existing RDMA operations either do not support multicast primitives with zero-CPU overhead (i.e., one-sided) or do multicast with a non-negligible amount of CPU cycles on the receiver end (i.e., two-sided) [14]. As two-sided operations cannot be performed without the CPU, the only way to enable the efficient zero-CPU RDMA multicast is thus through one-sided operations.

Note that one-sided operations require RKEY and RADDR parameters per receiver. Therefore, in multicast, there will be N different combinations of connection parameters <QPN, PSN, RKEY, RADDR> for N receivers, which would be difficult to be encoded in the RDMA headers. Even if they are encoded, existing RDMA transport hardware found on RNICs is fixed-function and cannot be modified to add custom functionality. This hinders the possibility of introducing innovations to the underlying RDMA transport hardware to enable one-sided, zero-CPU RDMA multicast.

To that end, we propose a framework that abstracts one-sided, zero-CPU RDMA multicast from the underlying RDMA transport hardware. Specifically, we refer to one-sided RDMA multicast as the one-sided WRITE operation with multicast in our subsequent discussions[1]. The core idea is to leverage

[1]RDMA READ and ATOMIC operations do not fit the context of multicast.

ANCS '21, December 13–16, 2021, Layfette, IN, USA

ACM ISBN 978-1-4503-9168-9/21/12.
https://doi.org/10.1145/3493425.3502766

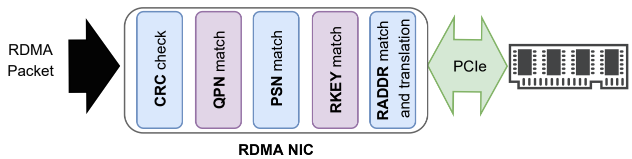

Figure 1: The RDMA processing pipeline.

the on-path P4 programmable hardware as the intermediate hardware shim layer to manipulate the RDMA packet headers to support one-sided RDMA multicast at line rate.

In §2, we first analyze how RNICs process one-sided RDMA packets. Then, we present a mechanism that abstracts one-sided RDMA multicast from the RNICs' transport hardware using P4 programmable hardware (e.g., switches and Smart-NICs) in §3. Lastly, in §4, we present our prototypes on the Intel Tofino ASIC-based programmable switches [12] and Mellanox BlueField2 DPU SmartNICs [21]. This work extends the one-sided RDMA multicast mechanism outlined in [24] and takes the next step in making one-sided, zero-CPU RDMA multicast suitable for general applications.

2 DISSECTING RDMA

Here, we outline the key requirements to abstract one-sided RDMA multicast from the RNICs. We focus on RoCEv2 [1] which is commonly adopted in data centers [9, 10].

2.1 The RDMA Pipeline

Fig. 1 depicts the processing pipeline in RDMA transports. Firstly, whenever a packet arrives, the RNIC verifies the checksums (e.g., IP checksum and invariant CRC32 checksum), then discards corrupted RDMA packets. Next, the destination QP number (QPN) is checked for whether the particular RDMA connection exists or not. Typically, one QP corresponds to a single instance of a one-sided operation at a time. Subsequently, the RNIC inspects the packet sequence number (PSN) to identify out-of-order or duplicate packets. Depending on the API used for the RDMA connection setup, the starting PSN can either be randomly assigned (`librdmacm`) or manually assigned (`libibverbs`) [22]. Lastly, the remote key's (RKEY) validity is scrutinized. The RKEY plays the role of the pointer to the memory region for use. Provided that the remote virtual address (RADDR) falls within the allocated memory region corresponding to the RKEY, the RADDR can be translated to the actual memory address, accessible directly via the PCIe bus.

2.2 Key Takeaways

As long as an RDMA packet has the correct parameters `<QPN, PSN, RKEY, RADDR>` matching an existing connection, it will be accepted by the RNIC's RDMA transport hardware to perform the corresponding RDMA operation [3, 22, 24]. Thus, one-sided RDMA multicast to a group of *N* receivers is possible if the on-path programmable network hardware can modify the multicasted RDMA packets with the "correct" parameters, i.e., <QPN, PSN, RKEY, RADDR> tuples, prior to reaching the RNICs. The top-of-the-rack (ToR) programmable switches connected to RNICs or the programmable packet processing pipelines within the RNICs [16, 21] are perfect vantage points to perform the modifications.

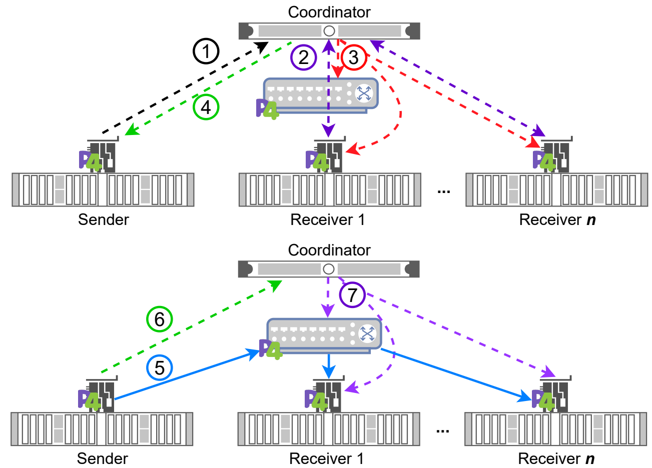

Figure 2: Mechanism for one-sided RDMA multicast

3 ONE-SIDED RDMA MULTICAST

Next, we discuss a mechanism leveraging P4 programmable switches and/or programmable RNICs to enable one-sided RDMA multicast. We introduce a central coordinator that acts as the proxy for the RDMA connection setup between the multicast group receivers and the sender. We illustrate the mechanism in Fig. 2. Multicast routing and group membership mechanisms are beyond the scope of this discussion.

3.1 Overview

First, the multicast sender initiates a multicast transfer request with the coordinator (step ①). Then, the coordinator performs RDMA connection setup with the group of receivers, acquires the necessary connection parameters (i.e., `<QPN, PSN, RKEY, RADDR>`) from every receiver of that group (step ②). The coordinator subsequently converts the connection parameters into match-action rules that perform RDMA header modifications and installs them into the programmable switch and/or the RNICs' match-action tables (step ③). Next, the coordinator signals the sender that it is "clear to send" (step ④). Upon receiving the "ready-to-send" signal from the coordinator, the sender proceeds with the multicast transfer (step ⑤). Once the transfer is completed, the sender notifies the coordinator (step ⑥) on the transfer completion. Finally, the coordinator teardowns the relevant match-action rules on the programmable switch and/or programmable RNICs (step ⑦). This completes the one-sided RDMA multicast process. For both the sender and receivers' RNIC, the one-sided RDMA multicast operation appears as if it is a one-sided RDMA `WRITE` in unicast.

3.2 Design Considerations

Here, we discuss the design considerations for programmable switches and programmable RNICs to enable the one-sided RDMA multicast mechanism.

3.2.1 Programmable Switches. First, we consider only using programmable switches for RDMA header mapping. The centralized coordinator configures the programmable switches with the mapping rules (step ③) which are derived from RDMA connection parameters collected from the receivers. Unfortunately, while using only programmable switches is feasible to support the mechanism, it may not be scalable as the available H/W resources (e.g., SRAM) comes at a premium [15]. For example, given a multicast group of N receivers, N RDMA header mapping rules have to be created. The ever-changing data center workloads [23] make it challenging for the switch to support such operations at scale.

3.2.2 Programmable RNICs. For better scalability, we can exploit the programmable RNICs to perform RDMA header mappings *before* handing over the packets to RDMA transport hardware. Specifically, the programmable pipeline in the RNICs can be either ARM cores [21], dedicated ASICs [16], or FPGAs [30] that come with abundant memory resources. Here, the programmable switches only perform multicast.

In practice, deployment does not have to choose one approach over another. Rather, programmable switches and RNICs can complement each other to support one-sided RDMA multicast.

4 PROTOTYPING

We prototype our proposed mechanisms to support one-sided RDMA multicast on both P4 programmable switch and P4 programmable RNIC.

4.0.1 Programmable Switch. Our local testbed consists of two x86 servers equipped with dual-port 25Gbps Mellanox ConnectX-5 RNICs. The ports are connected to an Intel Tofino [12] ASIC-based EdgeCore Wedge100BF-32X programmable switch. Using the four ports, we emulate a topology with one multicast sender and three receivers through Linux network namespaces.

We implement our prototype in $P4_{16}$ [4] in ~500 LoC. Apart from the match-action tables that are used to perform RDMA header field mapping, two separate register arrays are used to keep track of the PSN and RADDR offsets. Our prototype only supports up to 64B payloads as the switch has to parse and compute the invariant CRC32 checksum over a variety of header fields (including the data payload) that is at the tail of the packet [1]. Alternatively, checksum verification on the RNICs can be disabled [24] to support longer payloads but it may not be possible on most RNICs [6]. Next, we adapt the implementation in [2] to perform the initial RDMA connection setup with the receivers. The gathered connection parameters `<QPN, PSN, RKEY, RADDR>` are then configured into the switch. Lastly, we use `scapy` to craft RDMA `WRITE` packets from the sender.

Verification. We carry out multiple trials by multicasting 128MB file chunks from the sender to a group of 3 receivers. We verify that all receivers received the exact file chunks sent implying the correctness of the prototype implementation.

4.0.2 Programmable RNICs. In the absence of programmable RNICs on our local testbed, we perform the evaluations on our cloud testbed on CloudLab [7] equipped with two bare metal `r7525` servers equipped with Mellanox Bluefield2 DPU SmartNICs which are RDMA capable and has a P4 programmable packet processing engine that consists of 16 ARM cores and 32GB of memory. Because official P4 support are yet to be available [20], as a workaround, we implement our prototype in $P4_{16}$ for the behavioral model (BMv2) in ~400 LoC. Here, the RDMA header mapping is done by the ARM cores on the receiver's RNIC. We verify the prototype's correctness similar to §4.0.1 between the two end hosts.

5 FUTURE DIRECTIONS

We present a mechanism to augment existing fixed-function RDMA transport hardware with P4 programmable hardware to support one-sided RDMA multicast. While we envision that the framework can benefit various group communication-heavy applications, many challenges remains to be tackled in making one-sided RDMA multicast practical. How multicast interacts with PFC-enabled lossless Ethernet fabrics remain to be answered [10]. Furthermore, to ensure reliable communication, multicast congestion control algorithms for one-sided RDMA multicast need to be explored alongside mechanisms to handle packet retransmissions. Finally, scalable multicast routing and group membership mechanisms [13, 25] have to be integrated with the framework and evaluated.

ACKNOWLEDGMENTS

We thank the reviewers for their feedback. We would also like to thank Levente Csikor and Jialin Li for their comments and suggestions on the poster. This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 2 (Grant Number: MOE2019-T2-2-134).

REFERENCES

[1] Infiniband Trade Association. 2014. Annex A17: RoCEv2. https://cw.infinibandta.org/document/dl/7781 [Accessed: Oct 2021].

[2] Tarick Bedeir. 2010. *Rdma read and write with ib verbs*. Technical Report. Technical report, HPC Advisory Council, 2010. URL: https://www.hpcadvisorycouncil.com/pdf/rdma-read-and-write-with-ib-verbs.pdf.

[3] Rutger Beltman, Silke Knossen, Joseph Hill, and Paola Grosso. 2020. Using P4 and RDMA to collect telemetry data. In *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. 1–9.

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95.

[5] Jiaxin Cao, Chuanxiong Guo, Guohan Lu, Yongqiang Xiong, Yixin Zheng, Yongguang Zhang, Yibo Zhu, Chen Chen, and Ye Tian. 2013. Datacast: A Scalable and Efficient Reliable Group Data Delivery Service for Data Centers. *IEEE Journal on Selected Areas in Communications* 31, 12 (2013), 2632–2645.

[6] Mellanox Community. 2019. How to disable ICRC validation with RoCE v2? https://community.mellanox.com/s/question/0D51T00007A2M1fSAF/how-to-disable-icrc-validation-with-roce-v2 [Accessed: Oct 2021].

[7] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. 1–14.

[8] Linux Foundation. [n. d.]. Data Plane Development Kit (DPDK). http://www.dpdk.org [Accessed: Oct 2021].

[9] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, Fei Feng, Yan Zhuang, Fan Liu, Pan Liu, Xingkui Liu, Zhongjie Wu, Junping Wu, Zheng Cao, Chen Tian, Jinbo Wu, Jiaji Zhu, Haiyong Wang, Dennis Cai, and Jiesheng Wu. 2021. When Cloud Storage Meets RDMA. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 519–533.

[10] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over Commodity Ethernet at Scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 202–215.

[11] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. 2018. The express data path: Fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th international conference on emerging networking experiments and technologies*. 54–66.

[12] Intel. [n. d.]. Intel® Tofino™ Programmable Ethernet Switch ASIC. https://www.intel.com/content/www/xa/en/products/network-io/programmable-ethernet-switch/tofino-series.html [Accessed: Oct 2021].

[13] Theo Jepsen, Ali Fattaholmanan, Masoud Moshref, Nate Foster, Antonio Carzaniga, and Robert Soulé. 2020. Forwarding and Routing with Packet Subscriptions. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*. 282–294.

[14] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2016. Design Guidelines for High Performance RDMA Systems. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 437–450.

[15] Xin Zhe Khooi, Levente Csikor, Jialin Li, Min Suk Kang, and Dinil Mon Divakaran. 2021. Revisiting Heavy-Hitter Detection on Commodity Programmable Switches. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. 79–87.

[16] Patricia Kummrow. 2021. The IPU: A new, strategic resource for Cloud Service Providers. https://itpeernetwork.intel.com/ipu-cloud/ [Accessed: Oct 2021].

[17] Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, and Dan R. K. Ports. 2016. Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 467–483.

[18] Youyou Lu, Jiwu Shu, Youmin Chen, and Tao Li. 2017. Octopus: an RDMA-enabled Distributed Persistent Memory File System. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA, 773–785.

[19] Mike McBride and Olufemi Komolafe. 2020. *Multicast in the Data Center Overview*. Internet-Draft draft-ietf-mboned-dc-deploy-09. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-mboned-dc-deploy-09 Work in Progress.

[20] NVIDIA. [n. d.]. DOCA SDK. https://developer.nvidia.com/networking/doca [Accessed: Oct 2021].

[21] NVIDIA. [n. d.]. Mellanox BlueField2 DPU SmartNICs. https://store.mellanox.com/categories/dpu.html [Accessed: Oct 2021].

[22] Benjamin Rothenberger, Konstantin Taranov, Adrian Perrig, and Torsten Hoefler. 2021. ReDMArk: Bypassing RDMA Security Mechanisms. In *30th USENIX Security Symposium (USENIX Security 21)*. 4277–4292.

[23] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 123–137.

[24] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtarik. 2021. Scaling Distributed Machine Learning with In-Network Aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 785–808.

[25] Muhammad Shahbaz, Lalith Suresh, Jennifer Rexford, Nick Feamster, Ori Rottenstreich, and Mukesh Hira. 2019. Elmo: Source Routed Multicast for Public Clouds. In *Proceedings of the ACM Special Interest Group on Data Communication*. 458–471.

[26] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. 1–10.

[27] Xiaoye Steven Sun, Yiting Xia, Simbarashe Dzinamarira, Xin Sunny Huang, Dingming Wu, and TS Eugene Ng. 2018. Republic: Data multicast meets hybrid rack-level interconnections in data center. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 77–87.

[28] Xingda Wei, Rong Chen, and Haibo Chen. 2020. Fast RDMA-based Ordered Key-Value Store using Remote Learned Cache. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 117–135.

[29] Yiting Xia, T. S. Eugene Ng, and Xiaoye Steven Sun. 2015. Blast: Accelerating high-performance data analytics applications by optical multicast. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 1930–1938.

[30] Xilinx. [n. d.]. Alveo SN1000 SmartNICs. https://www.xilinx.com/applications/data-center/network-acceleration/alveo-sn1000.html [Accessed: Oct 2021].